Flush+Reload between Docker Containers

CPSC-6240 System Admin and Security
Group Project Proposal

Instructor: Dr. Jeffrey Alan Young
Students: Samuil Orlioglu and Anvitha Yerneni

Due: April 11, 2023
Submitted: May 3, 2023

1 Introduction

With the adoption of computer systems in world, computer security has been rising as a research interest. Many security threats try to penetrate a secured system directly. In recent years, researchers have taken interest in a class of attacks that are stealthier than traditional attacks. This class of attacks, dubbed side-channel attacks exploit meta-information of the system's operations. Cache side-channel attacks exploit CPU cache to extract or even disrupt the computer's operations. Flush+Reload[1] is one such attack that can extract the information about the processes running on a system. In this project, we investigate the efficacy of Flush+Reload between process running on separate Docker containers.

2 Background

Modern CPU processors have incorporated cache memory to assist in retrieving frequently accessed memory pages. Cached data allows the CPU to avoid

fetching data directly from main memory. This is a benefit because directly accessing the main memory is a relatively slow process and inevitably creates CPU idle time. Most computer architecture have a CPU with multiple cores. Most of these systems have a 3-level cache system in which cache level one and two are not shared between cores. However, level three cache(also known as last level cache) is commonly shared between all cores.

3 Flush+Reload

Flush+Reload is a side channel attack that allows an attacking process to spy and leak data from a victim process by exploiting the shared last level cache. Commonly, this attack monitors a shared memory address and it's usage in a concurrent victim process. Flush+Reload continually flushes the entire cache

and waits for the victim process to access the shared memory address. Then, the attacker accesses the same shared memory address and measures the access time. If the access time is shorter than some specified threshold, then the victim has indeed accessed the shared library. And conversely, if the attacker's access time is longer than the threshold, then the victim process has not accessed the shared library.

4 Docker

Docker is a containerization technology that allows a user to create "containers": isolated virtualized environments that runs software. These containers can be deployed in the cloud as services for users. However, given that these containers run in parallel of an underlying physical system, it may be possible that a malicious actor can deploy a container with the purpose of spying on other containers running on the system. Since Flush+Reload attacks work even between virtual machines, it is logical to assume that Flush+Reload will work between containers, thereby posing a risk to many applications. Flush+Reload can be used to steal private keys, log network activity, work as a keylogger and many more malignant possibilities.

5 Proposal

We propose to research the leaking capabilities of Flush+Reload between Ubuntu Docker containers. Our research path is

1. Develop a simple shared library that that attacker and victim process will access.
2. Develop a victim container with the victim process that uses the aforementioned shared library.
3. Develop an attacker container that will utilize the Flush+Reload techniques to spy on the victim process in the victim container.

The reason for developing our own shared library over using an existing shared library is to avoid system level noise in the attack. However, if our shared library does not show to be promising, we will use an existing shared library such as `libc.so`.

We will develop the victim process to access the shared library in a pattern of accesses. Then, when the attacking process logs these accesses, we will compare the accesses reported by the attacker to the actual accesses to determine the efficacy of our approach.

5.1 Evaluation Methodology

We will develop the victim process to access the shared library in a pattern of accesses. Then, when the attacking process logs these accesses, we will compare

the accesses reported by the attacker to the actual accesses to determine the efficacy of our approach.

5.2 Tools and Software

We will use Docker as the containerization technology. We will create two Ubuntu containers that will act as attacker and victim. The victim process will be compiled from a c script that utilizes the common shared library. The attacking process will be compiled from a c script and will use inline assembly to flush the CPU cache and the shared library as the shared memory address between the victim and attacker.

5.3 Caveats

Modern Linux systems aim to prevent these types of attacks with Address Space Layout Randomization (ASLR). In our research, we will turn off this feature in order to execute the attack. However, if we have time at the end of the research, we would like to look into how Flush+Reload can be conducted on a system with address space layout randomization, but that is not our primary goal.

References

[1] Yuval Yarom and Katrina Falkner. 2014. FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, USA, 719–732.